AI ASSISTED CODING LAB

ASSIGNMENT 1.2

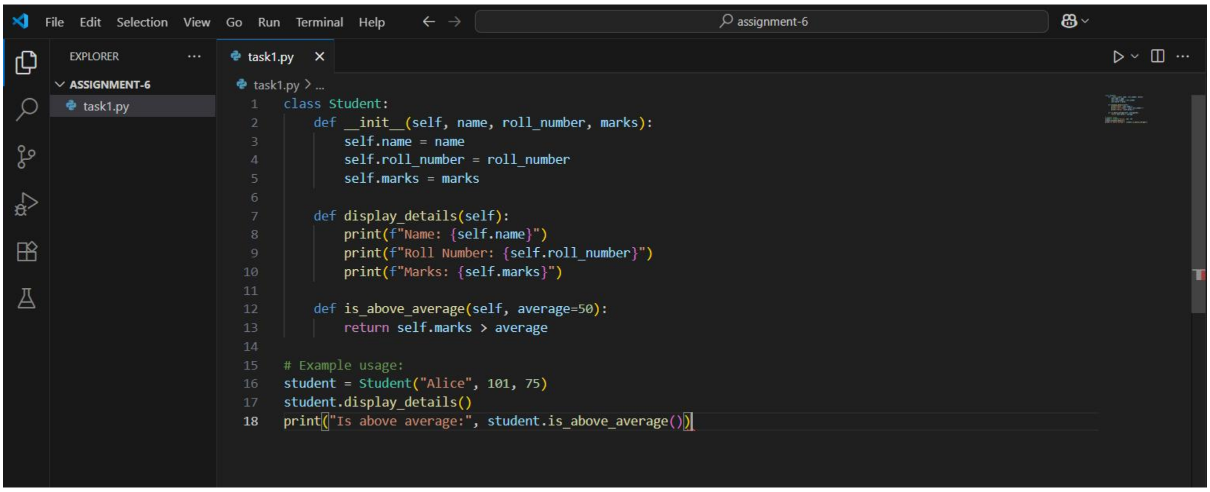
ENROLLMENT NO :2503A51L20

BATCH NO: 19

NAME: Guangsinlung Phaomei

TASK DESCRIPTION 1: Start a Python class named Student with attributes name, roll_number, and marks. Prompt GitHub Copilot to complete methods for displaying details and checking if marks are above average.

PROMPT 1: Generate a program that Start a Python class named Student with attributes name, roll_number, and marks. Add methods to display the student details and to check if the marks are above average.

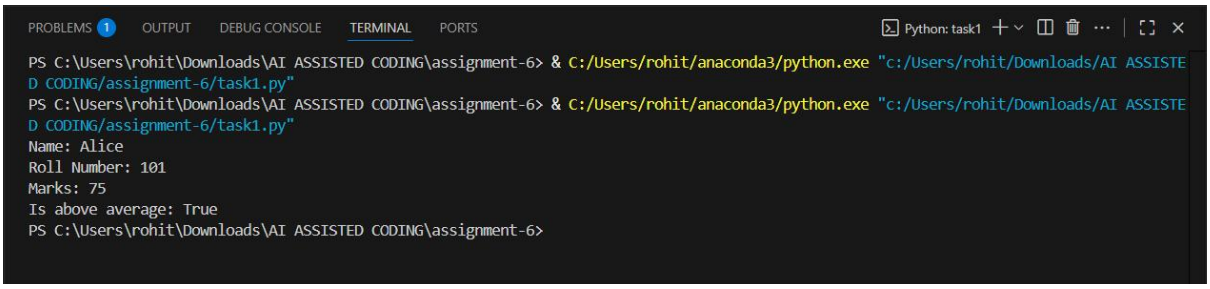


```python
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

    def is_above_average(self, average=50):
        return self.marks > average

# Example usage:
student = Student("Alice", 101, 75)
student.display_details()
print("Is above average:", student.is_above_average())
```

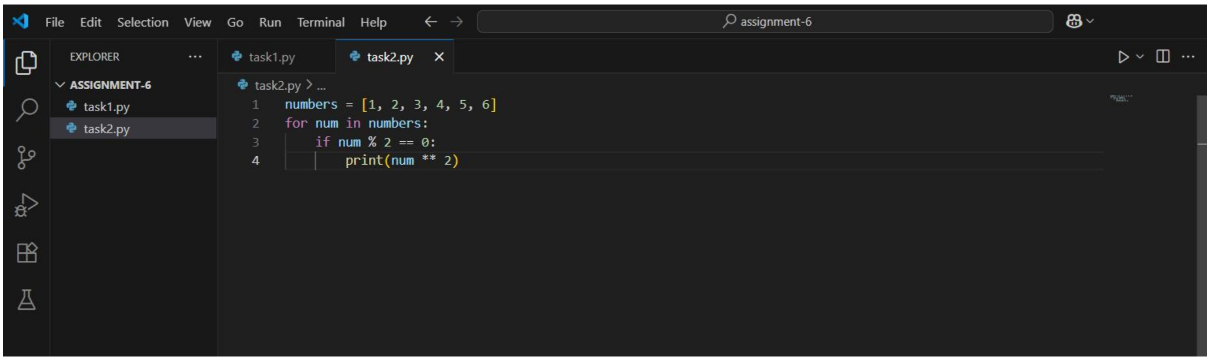


```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Downloads/AI ASSISTE
D CODING/assignment-6/task1.py"
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Downloads/AI ASSISTE
D CODING/assignment-6/task1.py"
Name: Alice
Roll Number: 101
Marks: 75
Is above average: True
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

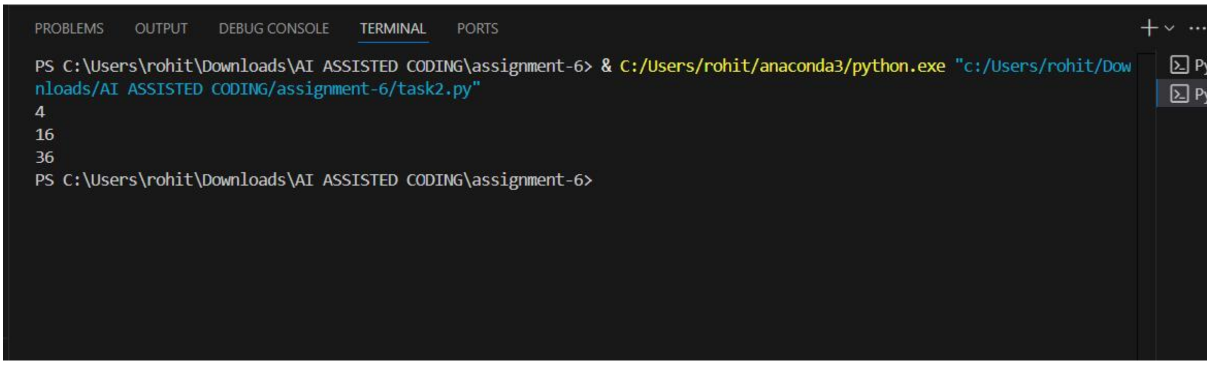
OBSERVATION: In this task, a Python class named Student was created with the attributes name, roll_number, and marks. Using GitHub Copilot, methods were generated to display the details of the student and to check whether the marks are above average. The prompt given guided Copilot to complete the class by adding the required methods. This demonstrates how AI-assisted coding tools can help in reducing development effort and improving productivity by quickly generating correct and structured code.

TASK DESCRIPTION 2: Write the first two lines of a for loop to iterate through a list of numbers. Use a comment prompt to let Copilot suggest how to calculate and print the square of even numbers only.

PROMPT 1: Generate a code to Create a list of numbers and write the first two lines of a for loop to iterate through that list. Add a comment inside the loop asking Copilot to suggest code that calculates and prints the square of even numbers only.



```python
numbers = [1, 2, 3, 4, 5, 6]
for num in numbers:
    if num % 2 == 0:
        print(num ** 2)
```

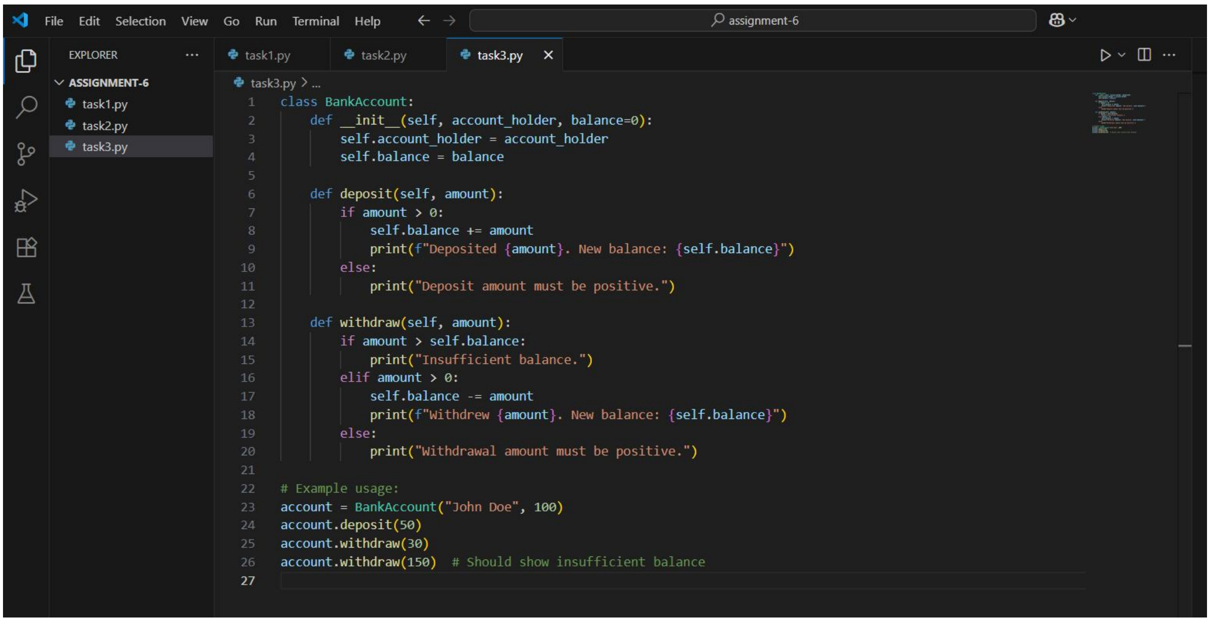


```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task2.py"
4
16
36
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: In this task, a list of numbers was created and the first two lines of a for loop were written to iterate through the list. A comment prompt was added inside the loop to guide GitHub Copilot in generating the code that calculates and prints the square of even numbers only. This showed how Copilot can interpret natural language comments and suggest the required logic, making the coding process more efficient and reducing manual effort.

TASK DESCRIPTION 3: Create a class called Bank Account with attributes account_holder and balance. Use Copilot to complete methods for deposit (), withdraw (), and check for insufficient balance.

PROMPT 1: Generate a python code to Create a Python class named Bank Account with attributes account_holder and balance. Ask Copilot to generate the methods deposit () and withdraw (), and to include a check that prevents withdrawal when the balance is insufficient.



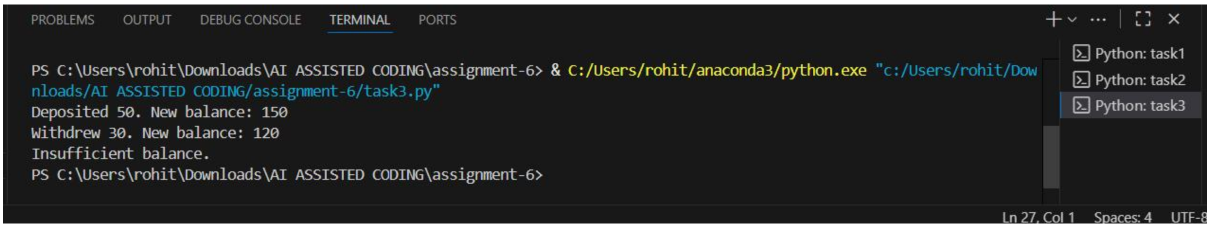
```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance.")
        elif amount > 0:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
        else:
            print("Withdrawal amount must be positive.")

# Example usage:
account = BankAccount("John Doe", 100)
account.deposit(50)
account.withdraw(30)
account.withdraw(150)  # Should show insufficient balance

```

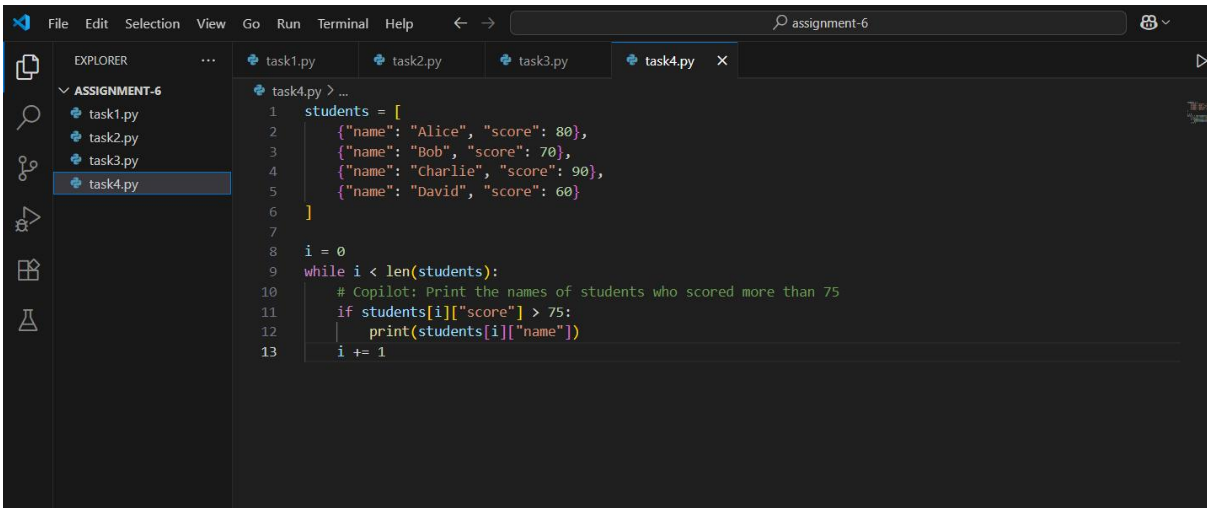


```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task3.py"
Deposited 50. New balance: 150
Withdrew 30. New balance: 120
Insufficient balance.
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: In this task, a Python class named Bank Account was created with the attributes account_holder and balance. Using the given prompt, GitHub Copilot was guided to generate the methods deposit () and withdraw (). Additionally, Copilot included a condition to check for insufficient balance before allowing withdrawal. This demonstrates how Copilot can automate the creation of commonly used functionalities in object-oriented programming, ensuring correctness while saving time for the programmer.

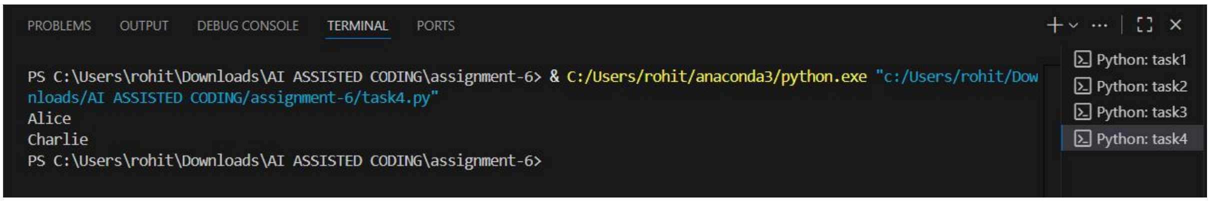
TASK DESCRIPTION 4: Define a list of student dictionaries with keys name and score. Ask Copilot to write a while loop to print the names of students who scored more than 75.

PROMPT 1: Generate a code to Define a list of student dictionaries where each dictionary contains the keys name and score. Ask Copilot to generate a while loop that prints the names of all students whose score is greater than 75.



```python
students = [
    {"name": "Alice", "score": 80},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 90},
    {"name": "David", "score": 60}
]

i = 0
while i < len(students):
    # Copilot: Print the names of students who scored more than 75
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

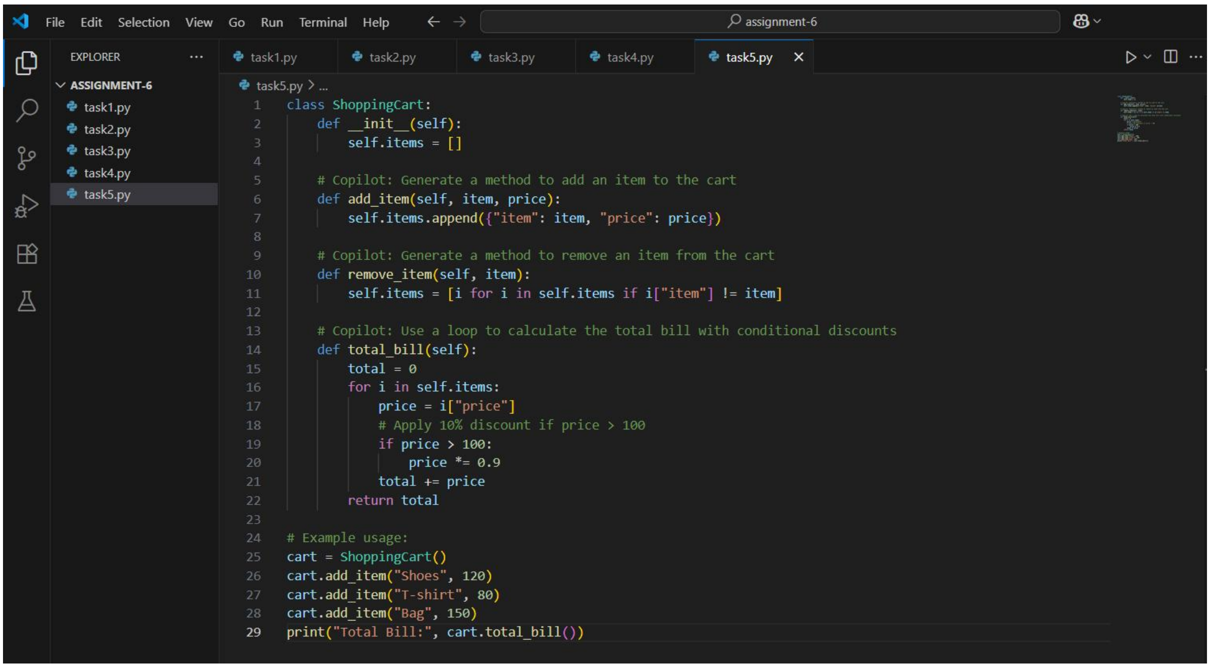


```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task4.py"
Alice
Charlie
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: In this task, a list of student dictionaries was defined with the keys name and score. GitHub Copilot was then prompted to generate a while loop that prints the names of students who scored more than 75. This activity highlighted how Copilot can understand structured data and apply conditional logic within loops to filter and display specific information, thereby simplifying the coding process.

TASK DESCRIPTION 5: Begin writing a class Shopping Cart with an empty items list. Prompt Copilot to generate methods to add_item, remove_item, and use a loop to calculate the total bill using conditional discounts.

PROMPT 1: Generate a code that Begin writing a Python class named ShoppingCart with an empty items list. Ask Copilot to generate methods add_item () and remove_item (), and to include a loop that calculates the total bill with conditional discounts applied.



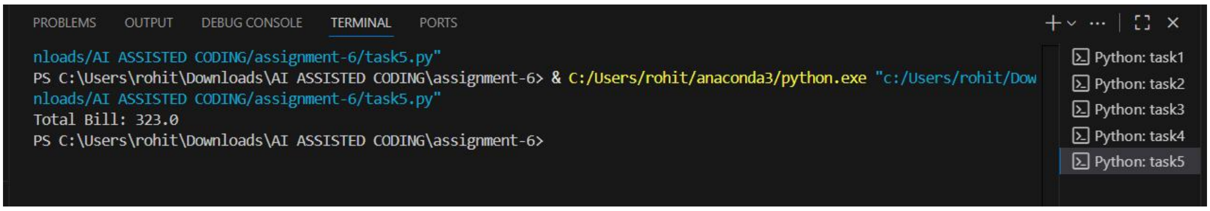
```python
class ShoppingCart:
    def __init__(self):
        self.items = []

    # Copilot: Generate a method to add an item to the cart
    def add_item(self, item, price):
        self.items.append({"item": item, "price": price})

    # Copilot: Generate a method to remove an item from the cart
    def remove_item(self, item):
        self.items = [i for i in self.items if i["item"] != item]

    # Copilot: Use a loop to calculate the total bill with conditional discounts
    def total_bill(self):
        total = 0
        for i in self.items:
            price = i["price"]
            # Apply 10% discount if price > 100
            if price > 100:
                price *= 0.9
            total += price
        return total

# Example usage:
cart = ShoppingCart()
cart.add_item("Shoes", 120)
cart.add_item("T-shirt", 80)
cart.add_item("Bag", 150)
print("Total Bill:", cart.total_bill())
```



```
nloads/AI ASSISTED CODING/assignment-6/task5.py"
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit/Dow
nloads/AI ASSISTED CODING/assignment-6/task5.py"
Total Bill: 323.0
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-6>
```

OBSERVATION: In this task, a Python class named ShoppingCart was created with an empty items list. GitHub Copilot was prompted to generate the methods add_item () and remove_item (), and to implement a loop that calculates the total bill while applying conditional discounts. This exercise demonstrated how Copilot can assist in developing object-oriented programs with dynamic operations on lists and applying business logic, making the coding process more efficient and less error-prone.